2021-09

CONCEPT DRIFT FOR DISCRETE-EVENT SIMULATION MODELING OF MANUFACTURING SYSTEMS

Ng, Wei Xiang

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CONCEPT DRIFT FOR DISCRETE-EVENT SIMULATION MODELING OF MANUFACTURING SYSTEMS

by

Wei Xiang Ng

September 2021

Thesis Advisor: Devaushi I. Singham
Second Reader: Michael P. Atkinson

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE CONCEPT DRIFT FOR DISCRETE-EVENT SIMULATION MODELING OF MANUFACTURING SYSTEMS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Wei Xiang Ng	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Many discrete-event dynamic systems face changing underlying conditions over time. Manufacturing processes are one example of systems that can be well modeled using discrete-event simulation. System changes include cyclical changes due to natural queueing behavior and changes due to random variability, either of which can involve significant structural changes to the system. For example, machines can degrade in performance, work shift schedules can change, or resources can be reallocated in a state-dependent manner. The ability to detect such changes is an important part of *concept drift*, which attempts to determine when a model may be outdated due to changing conditions. This research aims to formally integrate discrete-event simulation with methods of concept drift using a military vehicle manufacturing case study. We use simulation modeling to build a comprehensive model of this notional system and collect time series output under a variety of conditions. These output time series of queueing data are used to fit a recurrent neural network that can be used to identify state changes of interest. This work has broader implications in developing methods for detecting unusual activities using concept drift.

14. SUBJECT TERMS concept drift, discrete event simulation, manufacturing systems	**15. NUMBER OF PAGES** 83
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

CONCEPT DRIFT FOR DISCRETE-EVENT SIMULATION MODELING OF MANUFACTURING SYSTEMS

Wei Xiang Ng
Civilian, Singapore Technologies - Engineering Land Systems
MSE, National University of Singapore, 2020

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 2021

Approved by: Devaushi I. Singham
Advisor

Michael P. Atkinson
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Many discrete-event dynamic systems face changing underlying conditions over time. Manufacturing processes are one example of systems that can be well modeled using discrete-event simulation. System changes include cyclical changes due to natural queueing behavior and changes due to random variability, either of which can involve significant structural changes to the system. For example, machines can degrade in performance, work shift schedules can change, or resources can be reallocated in a state-dependent manner. The ability to detect such changes is an important part of *concept drift*, which attempts to determine when a model may be outdated due to changing conditions. This research aims to formally integrate discrete-event simulation with methods of concept drift using a military vehicle manufacturing case study. We use simulation modeling to build a comprehensive model of this notional system and collect time series output under a variety of conditions. These output time series of queueing data are used to fit a recurrent neural network that can be used to identify state changes of interest. This work has broader implications in developing methods for detecting unusual activities using concept drift.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CE	Concurrent Engineering
DOD	Department of Defense
DOE	Design of Experiments
EMD	Engineering and Manufacturing Development
FRP	Full Rate Production
LH	Latin Hypercube
LRIP	Low-Rate Initial Production
MRO	Maintenance Repair Overhaul
MSE	Mean Square Error
NARX	Nonlinear Autoregressive Exogenous
NOLH	Nearly Orthogonal Latin Hypercube
OLH	Orthogonal Latin Hypercube
RNN	Recurrent Neural Network
WIP	Work in Progress
WPS	Welding Procedure Specification

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The term *concept drift* denotes a significant change in system behavior that may cause a model to be outdated due to changing conditions. Within a manufacturing production setting, the system behavior changes can be represented as delays in the production line due to work station unavailability from equipment failure. When concept drift has occurred, we try to determine events of interest that caused the drift. In an armored vehicle manufacturing setting, a drift in the work station availability can have significant impact on the product delivery schedule, which can result in late delivery penalties if the product is not delivered on time as required by the contract.

To detect concept drift, we employ the use of a nonlinear autoregressive exogenous (NARX) algorithm. The NARX algorithm is a form of recurrent neural network (RNN) that can be used in modeling sequence data and producing predictive results. The NARX algorithm uses current and past values of the variable that we wish to (denoted as the primary time series) observe as well as exogenous data (denoted as the secondary time series). Exogenous data are externally determined but can influence or be influenced by the primary time series.

To produce data for input into the NARX algorithm, we developed a simulation model of a production system consisting of seven main work stations that process three different variants of an armored vehicle product. We use the simulation model to generate data for the following parameters: station capacity, queue length for the different stations, and quantity of each variant in the production system. Next, we create three scenarios where the station capacity fluctuates according to different rules. The first scenario is where the station capacity changes at deterministic fixed intervals. The second scenario is where the station capacity changes randomly at time intervals based on an exponential distribution. The third scenario is where the station capacity changes dynamically, depending on the current queue length of that station. We generate two different sets of data from the simulation run for each of the three scenarios to serve as the training and validation datasets. Subsequently, we try to predict the values of two different primary time

series and test the predictions using each scenario. The first primary time series uses the capacity of the station of interest as the primary time series input, while the queue length of each work station and the quantity of each variant in the system are set as the secondary time series input. The second primary time series uses the queue length of the station of interest as the primary time series input, while the capacity of the station of interest, queue lengths of the other work stations, and quantity of each variant in the system are set as the secondary time series input.

Once we compile data for input into the NARX algorithm, we tune the NARX parameters which determine the amount of past data used to predict future values. There are three key parameters that the NARX algorithm requires: the time lags used for prediction for the primary time series (variable of interest), time lags for the secondary time series (exogenous variables), and the number of hidden neurons in the layer of the RNN. Depending on the range of values we wish to test for these parameters, the number of runs required to determine the optimal values could be very large. Hence, we employ Nearly Orthogonal Latin Hypercube (NOLH) sampling to determine the three parameters and found that having three time lags for the primary and secondary time series as well as three hidden neurons in the layer worked well for our setting.

After we tune the NARX algorithm parameters, we evaluate the model's performance based on the validation mean square error (MSE) values for three different prediction interval time steps of one hour, four hours, and eight hours. Besides the fixed time steps, we also compute another set of results at times of discrete event changes which results in a variable time step. This additional case only considers the times when the state variable of interest changes and computes the MSE at these discrete change points. Besides the model performance evaluation, we explore alternative experiments to study how the NARX algorithm would perform under different settings. The alternative experiments include comparing the model's prediction against another time series that it was not meant to predict, and reducing the number of input variables to see if fewer variables are sufficient to produce an accurate prediction.

The general takeaways are that the NARX algorithm produce more accurate predictions when the time step is small and when there are fewer change-times for the variable of interest. The experiment that aims to predict the queue length when the station capacity changes at random time intervals had the highest MSE values (worst prediction) across the different time steps. We hypothesize that the NARX algorithm does not produce predictions that are as accurate in the random change scenarios compared to the deterministic and dynamic scenarios. Also, the queue length variable is more difficult to predict than the station capacity variable because it can take a larger range of values. We also found that generally, NARX delivers better predictions on models and scenarios it was trained on. Though in some cases, due to stochastic sampling error, models trained on one scenario work better on other types of scenarios. Lastly, we found in some cases that the number of input variables could be reduced and still result in the same quality of predictions. Hence, we hypothesize that the NARX algorithm can perform well even with reduced inputs as long as there is a relationship between the variables that the algorithm is able to identify.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to extend my deepest gratitude to Dr. Dashi Singham and Dr. Michael Atkinson for taking time out of their busy schedules to provide me with their invaluable advice and guidance in completing this thesis.

The completion of this thesis also would not have been possible without the support patience, and love from my wife, Hayley, who has been taking care of our two daughters, Yue En and Yue Jun, back in Singapore while I am focusing on this thesis and the rest of the academic requirements.

Lastly, I would like to thank all my friends who have supported and helped me along the way. Without them, I would not have been able to achieve what I have today. Thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The military armored vehicle manufacturing industry consists of production for various products like combat tanks and specialized weapons as well as providing Maintenance Repair Overhaul (MRO) services for the previously mentioned equipment. This industry has experienced strong growth in the last decade although recently there was a slight decline due to the COVID-19 pandemic affecting economies everywhere (IBISWorld, 2021). Enhancing the nation's defense competitive advantage remains as a key objective for many countries. Hence, governments around the world allocate a large proportion of their budget on defense; for example, in 2021, Singapore's government announced a defense budget of SGD 15.36 billion (USD 11.56 billion). This is approximately 15% of the total government spending for the year and is a 12.7% increase over the revised 2020 defense budget (Janes, 2021). The United States, which holds the title for the highest defense spending in the world, spent over USD $714 billion in FY 2020 and this value is expected to increase to USD $733 billion in FY 2021 (Government Accountability Office, 2021).

Within the U.S. military armored vehicle industry, approximately $3.5 billion dollars of revenue was generated by over 70 defense companies (Dun & Bradstreet First Research, 2021). Considering the large sums of money involved, the U.S. Department of Defense (DOD) implements an elaborate system acquisition process with a series of phases, milestones, and reviews from the start till the end.

While defense spending is on an upward trend which suggests larger revenues for the military vehicle manufacturing industry, keeping the production cost low is vital for the manufacturer to generate profits. Due to the nature of the product being labor intensive and raw material being costly, keeping the production costs low becomes significant in improving profit margins. Although the idea of manufacturing automation is frequently raised as a potential solution of reducing costs, the manufacturing method for military vehicles remains heavily reliant on manual labor. However, manpower planning and allocation can be tricky as the manufacturing processing time at the different stages can vary. When this is combined with delays in material arrival due to supplier issues or

equipment breakdown, determining the amount of manpower to allocate to a particular station to alleviate the loading to prevent bottlenecks in the production line can become complex very quickly.

A. THESIS MOTIVATION

> It takes many good deeds to build a good reputation, and only one bad one to lose it.
>
> —Benjamin Franklin

The term *concept drift* is defined as the changes in the system behavior which may cause an existing model to be outdated due to changing conditions (Zenisek et al., 2019). For this manufacturing application study, the system behavior changes can be represented as delays in the production line due to reduced station capacity from equipment failure. The station capacity determines whether the station is available for work processing or closed for maintenance. When a capacity reduction occurs, it causes the work in progress (WIP) to form queues at the affected station, resulting in a point of congestion within the production line. Beyond this scope, there can be broader implications in developing methods for detecting unusual activities or key changes in the system and mapping them to events of interest using concept drift in other settings.

With these implications in mind, the ability to predict the future state of the production line and act on it before delays occur becomes invaluable as there can be significant late delivery penalties if the product is not delivered on time as required by the contract. To compensate for the time lost, higher costs will be incurred from deploying additional manpower to complete the work activities. There can also be other intangible implications such as the loss of organization goodwill and reputational damage. The quote on reputation by Benjamin Franklin comes to mind on the amount of effort to build good reputation and how easily it is lost.

To overcome reputational damage, the organization must either work on its ability to meet expectations or to reduce customer's expectations by making lesser promises (Eccles et al., 2007). However, making lesser promises may cause the customer to lose interest as their expectations are not met. Therefore, the forward-looking approach of

improving the organization's capabilities should be taken instead. By using future production state prediction, we can determine when delays will occur from data that can be collected from the production floor and take the necessary actions to reduce the effects of the delay.

To predict the future state of the production line, we will use nonlinear autoregressive with exogenous inputs (NARX) models which is a form of recurrent neural network (RNN). RNN are a class of neural networks that are used in modeling sequence data and can produce predictive results for sequential data. As the state of the production model parameters changes dynamically over time, the RNN class is suitable for use here. We use NARX specifically as it uses the current and past data values of a primary time series as well as inputs from exogenous data which we define as the secondary time series. The exogenous data are externally determined, but may influence the primary time series indirectly. An example of the primary and secondary time series is provided in the next paragraph.

For our manufacturing setting, we use simulation to generate the data for a notional model of an armored vehicle production line based in Singapore. The production line supports the manufacturing of three types of vehicle variants, each of which requires different processing times at every work station. The data generated from the simulation that make up the time series are the amounts of WIPs queueing to enter a station for processing, quantity of each variant in the production system, and station capacity. The station capacity determines the number of WIPs that can be processed at any one time. For our thesis, we create two time series with the generated data. The first primary time series uses the capacity of the station of interest as the primary time series input, while the queue length of each work station and the quantity of each variant in the system are set as the secondary time series input. The second primary time series uses the queue length of the station of interest as the primary time series input, while the capacity of the station of interest, queue lengths of the other work stations, and quantity of each variant in the system are set as the secondary time series input.

This thesis explores the use of machine learning to study and predict concept drift that occurs in the manufacturing production system based on various inputs such as the queue length of the various processing stations that can be collected on the production floor. To represent concept drift in the production line, we develop three different scenarios where the work station capacity changes depending on certain conditions. When the work station capacity drops to zero, no work activities can be carried out at that station. The three scenarios we develop are: Deterministic, Random, and Dynamic. In the real world, the deterministic scenario can be compared to situations where the work station equipment is scheduled for maintenance and the maintenance duration is known. For the deterministic scenario, we set the capacity to change at fixed intervals. Next, the random scenario represents equipment breakdowns that can occur and the mean time between failures as well as the mean downtime duration is known to follow some type of random distribution. For the random scenario, we set the capacity change to follow an exponential distribution. As for the dynamic scenario, it can be described as some form of cost cutting measure where we only activate the work station when there are sufficient WIPs to carry out the work activities. For the dynamic scenario, we set the capacity value to increase when the queue length exceeds a certain threshold value and decrease when the queue goes back down. The settings implemented from these scenarios would change the production state causing queues to form when we set the work station capacity to reduce after certain conditions. The parameter details of the three scenarios are as follows:

- Deterministic – Station capacity changes at fixed intervals which we set the station up time as 200 hours and station down time as 80 hours.

- Random - Station capacity changes at intervals based on an exponential distribution centered around a mean of 200 hours for the station up time and an exponential distribution centered around a mean of 80 hours for the station down time.

- Dynamic – Station capacity changes based on the current queue length of the work station within the production line.

There are two goals for the three scenarios. The first goal is to predict whether the capacity has gone down at the station by observing the number in the queue for the various work stations and number of variants in the system. The second goal is the reverse of the first goal, whereby we wish to predict the number in the queue given that there is a change in the station capacity. Implementing these three scenarios can provide us with insight on the NARX algorithm performance by testing its capability in predicting the state of the production line when concept drift occurs. For example, when the work station's equipment breaks down, that station's capacity drops to zero and no work activities can be carried out. This would cause queues to build up with a possible floor space shortage and result in a non-optimal allocation of manpower resource. Having a model that can predict the queue build up in advance by monitoring system state variables will be useful to the production manager in planning the floor space and manpower resource allocation.

B. SCOPE

We develop two simulation models using Simio software: a notional model of an armored vehicle production line and a scaled down simpler version. The notional production model which consists of seven main workstations, is based on a fabrication plant layout that is used to manufacture a series of armored vehicles in Singapore. On the other hand, the scaled down version of the production model consists of three stations and is first used to test out the NARX algorithm and to fine tune its parameters. Three variations of both models are developed to test different scenarios of concept drift: deterministic, random, and dynamic.

We generate the station queue, quantity of WIPs in the system, and station capacity data using a Simio simulation run and feed the values to the NARX algorithm which is run using MATLAB. Besides the data input, the NARX algorithm also requires three key inputs: number of time lags for the primary time series, number of time lags for the secondary time series, and the number of hidden neuron layers.

Next, we use Nearly Orthogonal Latin Hypercubes (NOLH) to determine the robust values for the three NARX inputs by testing within a range of 1 to 10 for the time steps and 1 to 20 for the number of hidden neuron layers on the scaled down simpler version of the

production system model. NOLH is an efficient way of conducting design of experiments (DOE). Using NOLH, we can test a subset of the experiments and obtain a significant representation of the data. Compared to testing a full factorial design where all possible combinations of the parameters are considered, we can save time and resources by using a NOLH.

Subsequently, we generate the production model output for the number in the queue for the work stations, number of each variant in the system, station capacity of interest and input these state variable trajectories into the NARX algorithm. The algorithm output is then analyzed and we attempt to draw insight into the accuracy of the NARX algorithm's prediction on when the station capacity drops by comparing it to the simulated output. We also try different test experiments to evaluate the algorithm. These include varying the time step interval input, reducing the number of input variables, permutating the training and validation datasets used, and computing the measure of performance only at the times when the station capacity changes.

Reviewing the results, we find that the NARX algorithm generally performs well when the time step is small which means that the data input points are sampled closer to each other and when there are fewer change-times for the variable of interest. The random scenario-queue response variable combination had the highest MSE values across the different time steps. This is likely due to the queue variable having a larger range of values compared to the station capacity variable and that the NARX algorithm does not predict randomness as well as the other two scenarios. For the experiment of training the NARX model on one scenario and validating it on another scenario, we found that there were three categories of results. The first category are the ones that perform as expected where the validation data MSE had a larger value and order of magnitude compared to the training data MSE. The second category had no significant change in MSE between the training and validation which would suggest that the NARX algorithm is able to predict accurately for those scenario-response combinations. The third category are the combinations that had a lower validation MSE than the training MSE. We hypothesize that this is possibly due to stochastic sampling error, sampling bias, and variability in the MSE across multiple computations. Lastly in the reduced exogenous inputs case, we tested a few combinations

and the NARX algorithm was able to produce results similar to those when using all the available variables. Hence, we can hypothesize that the NARX algorithm can perform well even with reduced inputs as long as there is a relationship between the variables that the algorithm is able to identify.

To recap, the first goal of this study is to predict the station capacity at a target work station given the observed values for the station queue lengths and number of each variant in the production system. In the second goal, we want to predict the queue length of the target work station given the change in station capacity in addition to the parameters mentioned for the first goal. Hence, we set the primary time series to reflect the two goals we want to achieve respectively and repeat the result analysis in each of the three scenarios (deterministic, random, and dynamic).

C. THESIS OUTLINE

Chapter II reviews the relevant literature related to this thesis including, NARX, NOLH, simulation, queueing theory, the DOD system acquisition process and military vehicle manufacturing. Chapter III discusses the problem formulation, methodology, methods and tools used to develop the scaled down simulation model as well as the final neural network. In Chapter IV, we review the NARX output based on the notional production model data input, analyze the results and draw insights from it. Lastly, Chapter V concludes the thesis and provides possible improvements for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LITERATURE REVIEW

Chapter II, Section A reviews the DOD system acquisition process, and the general process of armored vehicle manufacturing. Section B discusses the use of simulation modeling and introduces Simio software. Section C reviews queueing theory and some of the research on the applications of queueing theory in manufacturing systems. Section D explores the background of NARX and various research work that has applied it. Lastly, Section E covers the background of NOLH and the advantages of using it in DOE.

A. DOD SYSTEM ACQUISITION PROCESS AND ARMORED VEHICLE MANUFACTURING

When an order is placed for military vehicles, the product goes through a rigorous acquisition process used by the DOD which consists of six phases and three key milestones. It is important to go through these phases as every order tends to be different as the user will want to customize the product to their specific operational needs. Figure 1 shows the DOD system acquisition process, the breakdown of the six phases and when the three milestones take place.

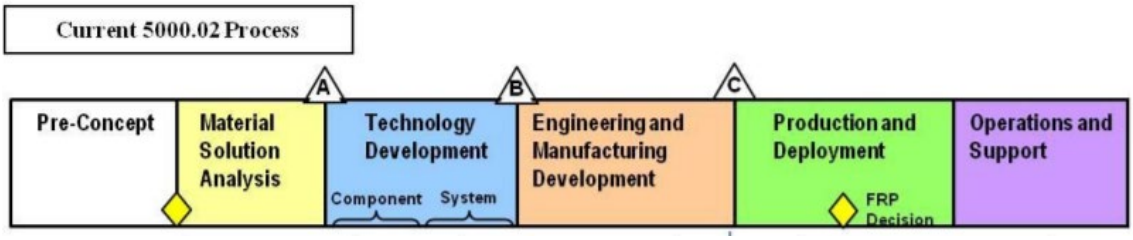

Figure 1. The six phases and three key milestones (A, B, C) of the DOD system acquisition process. Adapted from Defense Acquisition University (2012).

A team consisting of different department disciplines are required to support the entire acquisition process for an armored vehicle. The program manager acts as the main point of contact between the manufacturer and customer, engineering designers conceptualize the vehicle design that is able to meet the required operating specifications,

the finance team handles all the monetary matters, the resource planning & control department manages the flow of material into the production line, the supply chain management team source and purchase the required raw materials and parts, the quality assurance team provides the in-process inspections and final product acceptance, and lastly the production manufacturing team develop the processes and manpower allocation to build the vehicle.

Throughout the system acquisition phases, concurrent engineering (CE) takes place across all departments to ensure that the program runs smoothly. CE is a strategic tool for achieving industrial competitiveness by developing products faster, cheaper, and better by utilizing multi-function teams (Ghodous et al., 2006). For example, the production team will try to influence the engineering designers to design the product for ease in manufacturability and repeatability in the production process. Likewise in the other direction, the designers will try to push the boundaries of the manufacturing technologies and capabilities to produce new cutting-edge product designs. We will now briefly review what happens in each phase and milestone of the DOD system acquisition process.

The first phase is the pre-concept. The pre-concept's main focus is on an initial capabilities document that identifies and describes the user's projected mission needs in the context of the threat to resolve or the task to complete. In this phase, early commitments to specific system solutions are avoided and the requirements are defined broadly in operational capability terms. The goal in this phase is to understand the user's requirements.

The second phase is the material solution analysis phase. This phase covers the technology opportunities (prototypes, other program references) and resources (foreign and local). In this phase, alternative concepts such as cooperative opportunities with other countries or companies and the procurement or modification of allied systems are examined to see if the user's projected missions can be met. In general, modifying existing systems is simply more cost and time effective compared to developing a brand-new system. The goal in this phase is to identify and develop a solution that provides the best possible value over the system's lifecycle and is capable of meeting the user's operational

requirements. At this point, we enter the first Milestone A where the material solution and funding is approved for the technology development phase.

The third phase is the technology development phase. This phase focuses on reducing technology risk, determining which set of technologies are mature enough to be integrated into the system, and demonstrating that the critical elements in the technology are workable in the prototype. This applies to both the component and system level. We then enter Milestone B where the technology is demonstrated to function properly in the relevant working environment. The manufacturing risks are also identified and the system is deemed to be ready for production within an agreed time frame. Upon meeting these requirements, the project is approved and receives the full funding.

The fourth phase is the engineering and manufacturing development (EMD) phase. In this phase, the system functionality and interface are defined. Different levels of design review are conducted as well. The goal of this phase is to reduce the risk of the program failing, ensure that the designs are producible and affordable, certify that operations are supportable, and demonstrate system integration. While pursuing the goal, all the critical program information must be protected. Once the EMD phase is complete, we enter Milestone C where an assessment is conducted to evaluate that the system is operationally effective and ready for full rate production without significant risks.

The fifth phase is the production and deployment phase. This phase can be further splits into two subphases, low-rate initial production (LRIP) and full rate production (FRP). The LRIP is essentially to evaluate whether the manufacturing capability is adequate and efficient enough. The vehicles produced during LRIP are then used for operational tests and evaluations. During this phase the focus is on executing the manufacturing plan which reflects the design intent and ensures that the manufacturing processes are repeatable. There is also an emphasis on continuous improvement which is referred to as Kaizen in some organizations (Medinilla, 2016). The goal of manufacturing is to produce uniform products with no defects at the lowest possible cost. Once all stakeholders are satisfied with the results, we then transit into the FRP subphase where all of the remaining orders are produced.

The last phase is the operations and support phase which covers the MRO related activities including system modifications, sustainment management, and disposal at the end of product lifecycle.

B. SIMULATION MODELING

Simulation modeling allows us to create digital prototypes of real-world systems to test various scenarios and analyze the results to draw insights to predict performance in the real world. This also more time and cost efficient compared to performing the actual trials. Habchi (2003) lists several articles covering the use of simulation modeling for a variety of fields and the different approaches used to draw insights in the real world. This demonstrates the widespread acceptance of modeling and simulation in different fields.

For our production model, we use Simio software to build the simulation model. Simio software is a "multi-paradigm modeling tool" which is capable of integrating various simulation models into a single framework (Kelton et al. 2014, p. 4). It is also considered an object-oriented simulation software. Object-oriented simulation contains objects that interact with each other over the simulation runtime period (Joines & Roberts, 1998). For our setting, some examples of these objects are the work stations and the WIPs.

The software has animations that depict the physical product flow to help the user understand what is happening. We can vary the various logic inputs such as the interarrival rates and processing time at each server. The output generated can also be exported to Microsoft Excel and customized to produce a specific output to suit the nature of the problem that we wish to analyze.

C. QUEUEING THEORY

Queueing theory is a field of study that can be traced back to the early 1900s. It is a mathematical study of how queues are formed based on the waiting and service times of entities. The characteristics of queueing theory include Kendal's Notation for describing types of queue systems and Little's Law which states the relationship between the system parameters like number in system, queue length, interarrival time, and processing time (Kelton et al. 2014, p 19 - 28). There are further applications in the form of reneging and

balking where the customer in the queue leaves the current position to join another queue or leave the system completely. There are also many queueing scheduling polices. Some of these policies include first-come first-serve where the first customer in the queue gets served first like a typical queue in a fast-food restaurant setting, and priority serving where customers with higher priority gets served first.

For our armored vehicle production setting, the "customer" is the WIP that is waiting to be processed at the workstation. Although queueing theory is used to ensure the production system is stable during the process planning, queues can still form for many reasons such as equipment failure, scheduled equipment maintenance, or the amount of time required to perform the work activity at a particular work station deviates from the expected duration depending on the technician's capabilities. This can create bottlenecks on the production floor which is a major headache for production managers as internal floor space is scarce and the WIPs cannot be left in the open as the metal plates will rust. A quick fix solution is to rent storage space from another location, but this can be costly if it is not planned in advance. Therefore, the study of queueing theory can complement a simulation model of the production floor and help to manage the cost aspect to reduce the impact on the profit bottom line.

There is much research on queueing theory being applied to manufacturing systems. Govil and Fu (1999, p. 214) describes the "contributions and applications of queueing theory in discrete part manufacturing." Ghalehkhondabi and Suer (2018) uses queueing theory to perform an analysis on production line performance within a manufacturing framework that mixes make to order and make to stock strategies to reduce customer waiting time.

D. NARX NEURAL NETWORK

The NARX neural network is a robust class of dynamic RNN models that predicts future values of a time series based on its past values as well as current and past values from an exogenous input series (Bianchi et al., 2018). It is commonly used for modeling nonlinear dynamic systems. NARX network applications include being used as a predictor for future time series values and also being used for nonlinear filtering where the output is

a noise free version of the input values. The neural network can be expressed mathematically as shown in the equation below.

$$y(t) = f\left[y(t-1), y(t-2), \ldots, y(t-d), u(t-1), \ldots, u(t-d)\right] + \in (t)$$

The terms are defined as follows: y = primary time series, u = secondary time series, t = time steps, d = time-lag limit, f = nonlinear function describing the system behavior, $\in$ = error term. A graphical representation of the NARX neural network framework is shown in Figure 2. The NARX neural network framework graphic uses two additional terms, w is the number of layers and $\hat{y}$ is the predicted primary series output. The number of layers measure the network depth while the number of neurons in each layer measures how wide the network is.

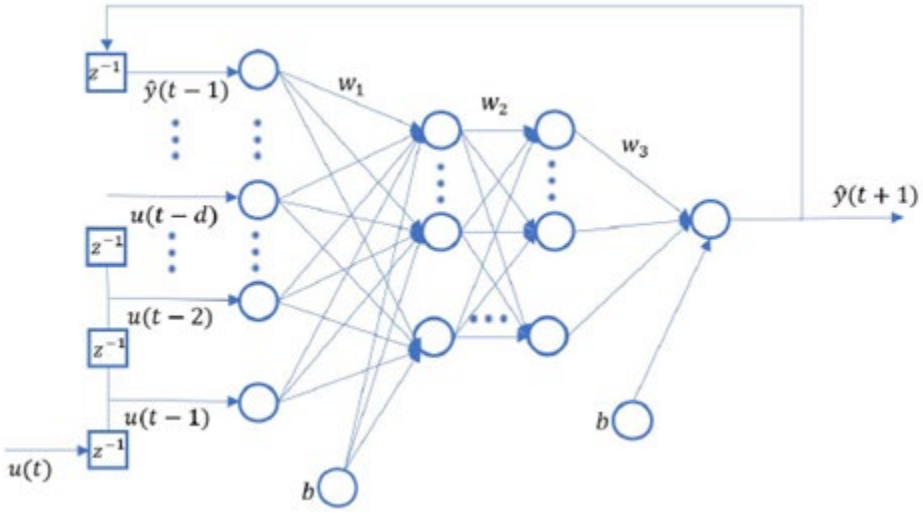

Figure 2. NARX Neural Network Framework. Adapted from Yildirim (2020)

Users define the number of time steps for the primary and secondary time series, as well as the number of hidden neuron layers in the NARX. The hidden neuron layers are the layers between the input and output layers. Each layer applies an activation function to the incoming input before it moves to the next layer. The main measure of performance of the network is the mean square error (MSE) values. MSE measures the average squared difference between the predicted and actual values in a time series. The mathematical

formula for MSE is shown below. The terms are defined as follows: *MSE* = mean square error, n = number of data points, Y_i = observed values, $\hat{Y}_i$ = predicted values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2$$

Lastly, we reviewed some of the research work relating to NARX neural networks. Yildirim et al. (2020) describes the approach of how the NARX neural network predicts the temporal resource levels using a time series that contains the past and current resource level values. In the paper, Yildirim et al. stated that NARX was used as the proposed method as it is more capable than "other alternatives such as linear parametric autoregressive or autoregressive moving-average models" (2020, p.4). Yildirim et al. used the trial-and-error method to determine the number of hidden neuron layers and training function parameters. As trial-and-error is not an efficient process especially when the number of combinations is large, we will streamline the parameter selection process using Nearly Orthogonal Latin Hypercube (NOLH) sampling. Another work by Xie et al. (2009) showed that the approach of using original NARX network architecture to predict time series outperforms standard neural network-based predictors such as a time delay neural network architecture when evaluating vibration data from a CO_2 compressor.

E. NEARLY ORTHOGONAL LATIN HYPERCUBE

The modern-day computer's capabilities have become significantly more powerful compared to its predecessors. With improved capabilities, users are able to analyze simulations of big and complex dataset experiment simulations which was previously not resource efficient. However, once the quantity of input parameters and number of experiments to conduct becomes extremely large it can take many days for computers to churn out results for full factorial experiments. Hence, there is a lot of research being conducted to explore means of extracting information efficiently from these large simulation runs and one resulting method is Latin Hypercube (LH) sampling. This is an efficient way of statistically generating near random quantitative parameter values from multiple dimensions as they have good space filling properties.

LH can be conceptualized as having an arbitrary number of dimensions and there is only one sample point in each axis-aligned hyperplane comprising it. In orthogonal sampling, the sample space is divided into subspaces equally and the sample points are picked simultaneously in a way where the subspaces have equal density. Therefore, orthogonal sampling provides a good representation of the dataset. Hernandez (2008) defines Orthogonal Latin Hypercube as a LH where there is no correlation between any input variable, while Nearly Orthogonal Latin Hypercube (NOLH) is defined as a LH variation where the maximum absolute pairwise correlation is not greater than 0.05 between any two input variables.

Viana (2016) describes in detail on how LH has become popular among DOE practitioners and the reasons for its popularity. There are several variations of LH using different techniques to extract information efficiently. LH has also become easy to implement due to the widespread availability in many computer simulation software (Hernandez et al., 2012).

III. PROBLEM FORMULATION AND METHODOLOGY

Chapter III discusses the manufacturing scenario for an armored vehicle production line, the various model assumptions, and how we model it in Simio. Next, we develop a scaled down model to generate input data and use NOLH to determine the NARX input parameters which are the number of time lags for the primary time series, number of time lags for the secondary time series, and the number of hidden neuron layers. After the NARX input parameter values are determined, we generate output from the original production model to feed into the NARX algorithm and derive insights from the results in Chapter IV.

A. SCENARIO DESCRIPTION

The armored vehicle series go through two sub-phases of production: fabrication and assembly. The first sub-phase is the fabrication phase where the hull, which is the "skeleton" of the armored vehicle, is formed by welding high strength metal plates together along with weld bosses. Figure 3 shows an image of a technician working on the hull structure with many bosses already welded. The weld boss is a reinforced ring for a threaded hole and used for installing components onto the vehicle. Figure 4 shows an example of a weld boss. After the hull welding is complete, it is sent for machining and subsequently routed to the assembly plant for the next phase. The second sub-phase is the assembly phase where the various components (e.g., computer systems, engine powerpack, drive system, ergonomic accessories like seats) are installed onto the hull to form the vehicle.

Figure 3. Technician working on a hull structure with many weld bosses
Source: Hallum (2021).

Figure 4. Weld boss with a threaded hole

For our manufacturing scenario we focus on the fabrication process, which takes place at a production plant based in Singapore. Figure 5 shows the overhead view of the production line. As seen from the image, floor space is a limited commodity and eliminating bottlenecks to prevent WIP buildup is essential.

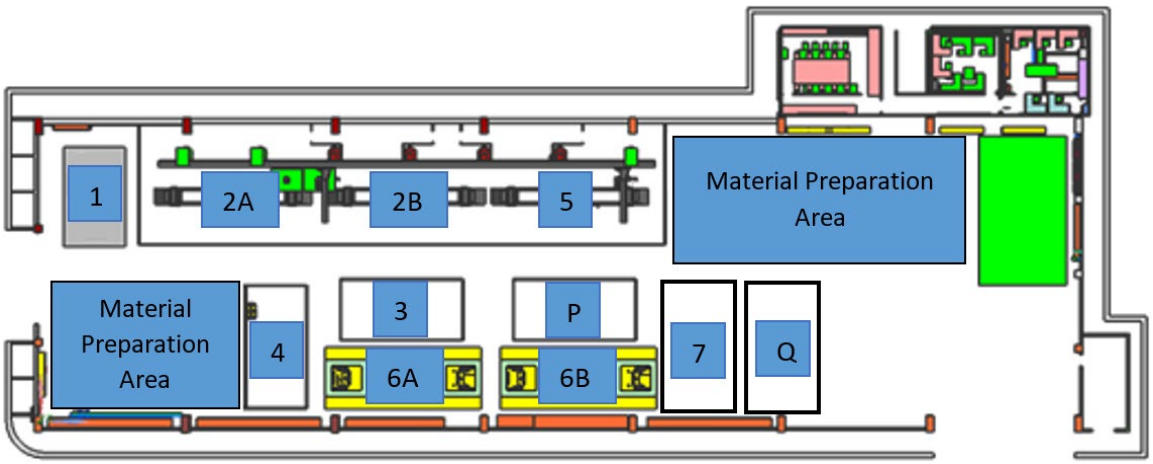

Figure 5. Armored vehicle production line (fabrication phase)

The stations are color-coded in blue with alpha-numeric labelling and the numbers indicate the flow of the WIP through the production line. The WIP goes through the stations sequentially. There are areas in the plant that are designated for material preparation activities such as degreasing the metal plates. There are also spaces designated for offices and loading/unloading activities. Any remaining unused floor space is used to hold the WIP and is considered as the queueing line for the various work stations. The descriptions for the respective work stations follow:

- Station 1: Setup of the bottom half of the hull. The plates are fitted using jigs and fixtures. This ensures that the product is made accurately according to the specifications and manufacturing repeatability. The plates are lifted with the aid of an overhead crane and tack-welded together manually to form the lower hull shape.

- Station 2: Full welding of all the seams in the lower hull. For our production line, there are two sets of Station 2 (labeled as 2A and 2B) as part of the original plant layout. The additional quantity is for line balancing as the time taken to process at Station 2 is significantly longer than the rest of the stations. In this station, the hull is set onto a manipulator that can rotate the WIP to the appropriate angle to perform the weld. The angle is dictated by the Welding Procedure Specification

(WPS). The full welding process can be done manually or automatically by robot welding.

- Station 3: Inspection of all the key weld seams after the lower hull welding is complete. A key weld seam is defined as a structural weld. An example of a non-key weld seam can be the weld seam of a stud or boss. This process is done manually by an inspector with the aid of non-destructive testing equipment.

- Station 4: Setup of the upper half of the hull. The processes that take place here are similar to that of Station 1. The fitting and tack welding processes are done manually.

- Station 5: Full welding of the reachable seams in the upper hull. As the roof of the hull is fitted, there can be accessibility issues when complying with the stated angle and position in the WPS. Hence whichever seams are reachable are welded at this station. The welding process can be done manually or automatically by robot welding.

- Station 6: Full welding of all the remaining unwelded seams in the hull. Similar to Station 2, there are two sets of Station 6 (labeled as 6A and 6B) as part of the original plant layout. The additional quantity is for line balancing as the time taken to process at Station 6 is significantly longer than the rest of the stations. The WIP is set onto a manipulator to rotate to the appropriate angle for the technician to perform the welding process. This process is done manually, as automated welding would have accessibility issues given the confined space within the hull.

- Station 7: Final inspection of all key and non-key weld seams after the upper hull welding is complete. Additionally, the hull distortion which results from the intensely high temperature from welding is also measured at this station.

- Station P: Rework station for any welding corrections after the lower hull welding is complete. After rework, inspection is conducted again to ensure that the required weld quality is achieved. This process is repeated until the requirement is met and the WIP moves on to the next station.

- Station Q: Rework station for any welding and/or distortion corrections after all the weld seams in the hull is complete. After rework, an inspection is conducted again to ensure that the required weld quality is achieved. This process is repeated until the requirement is met and the WIP moves on to the next manufacturing phase.

In our scenario, we need to produce three armored vehicle variants (Variant X, Variant Y, Variant Z) and they share the same production line and manpower. These variants have different processing times at each station as the designs are different. The ordering policy for the three variant's raw material is different as it is based on the quantity required and user demand rate. Therefore, the raw material arrival rate would differ between variants.

The three variants belong to the same product family with some differences between them depending on the intended mission capability. An example of a product with many variants is the Bionix Vehicle, which is developed by Singapore Technologies Engineering. Figure 6 shows different variants of the Bionix family: the Infantry Fighting Vehicle, Infantry Carrier Vehicle, Armored Recovery Vehicle, and the Armored Vehicle Launched Bridge.

Figure 6. Different Bionix variants. Top left: Infantry Fighting vehicle, top right: infantry carrier vehicle, bottom left: armored recovery vehicle, bottom right: armored vehicle launched bridge. Source: Army Technology (2021).

In our simulation model, the raw material average interarrival time and average processing at each station for each variant is stated in Table 1 below. The respective station processing time values were determined through a time study on the expected duration for the various work activities carried out at each station. The time taken for Station 3, Station 7, and the two rework stations have the same duration for all three variants. The rest of the stations have different processing times and the raw material interarrival time is different as well.

Table 1. Average station processing time and average raw material interarrival time for the respective variants

Parameter (mean values)	Variant A	Variant B	Variant C
Raw Material Interarrival Time	3 days	4 days	6 days
Station 1 Processing Time	14 hours	16 hours	18 hours
Station 2 Processing Time	44 hours	54 hours	48 hours
Station 3 Processing Time	4 hours		
Station 4 Processing Time	16 hours	24 hours	14 hours
Station 5 Processing Time	16 hours	20 hours	16 hours
Station 6 Processing Time	50 hours	56 hours	66 hours
Station 7 Processing Time	16 hours		
Rework Station P Processing Time	8 hours		
Rework Station Q Processing Time	16 hours		

The same rework crew rotates between the rework stations (Station P and Q) while the same quality inspector rotates between the inspection stations (Station 3 and 7) as the processing time at these stations are shorter compared to the seven work processing stations. This allows the production line to save manpower costs through better utilization of its staff. In the event where the rework crew is occupied at one of the rework stations and there is a WIP waiting for rework at another, the WIP will have to wait to be processed. The same logic is applied for the quality inspector and the inspection stations.

B. TRANSLATING THE REAL WORLD INTO A SIMULATION MODEL

Armored vehicle manufacturing contains many stochastic components as many of the work activities are performed manually, the rate of completion would vary between technicians. There are also unexpected events that can occur such as weld defects which

require additional rework, raw materials arriving late, and equipment failure. Hence, many processes within the production line can be considered as stochastic instead of deterministic. Therefore, by using simulation modeling software we are able to capture this stochastic detail of the real-world and draw insights from it.

We set the simulation runtime to 350 days which is relatively short compared to the real-world production line which typically takes a few years. One of the main differences with the real-world is that we set the Simio model to run 24 hours a day unlike the real world where the technicians typically work eight hours a day. Therefore, in our simulation one day is equivalent to three real world days. Using this logic, we set the total simulation run time to be three times shorter than the typical planned duration for the real-world production. Setting the production model to run 24 hours a day also resolves the issue of inflated waiting times being reported in Simio as the WIP will need to wait to be processed if we have breaks in between. By using this method, we can also reduce the amount of unnecessary data bloat.

We model the production line in Simio software as shown in Figure 7. For our simulation model, we used four types of objects. They are the Source, Server, Sink, and Resource objects. The Source, Server, and Sink nodes are represented by the gray box graphic in the model while the Resource objects are represented by an oval shape with a colored circle within it. Every node is linked by connectors to represent the product flow direction. We will now briefly describe these various objects.

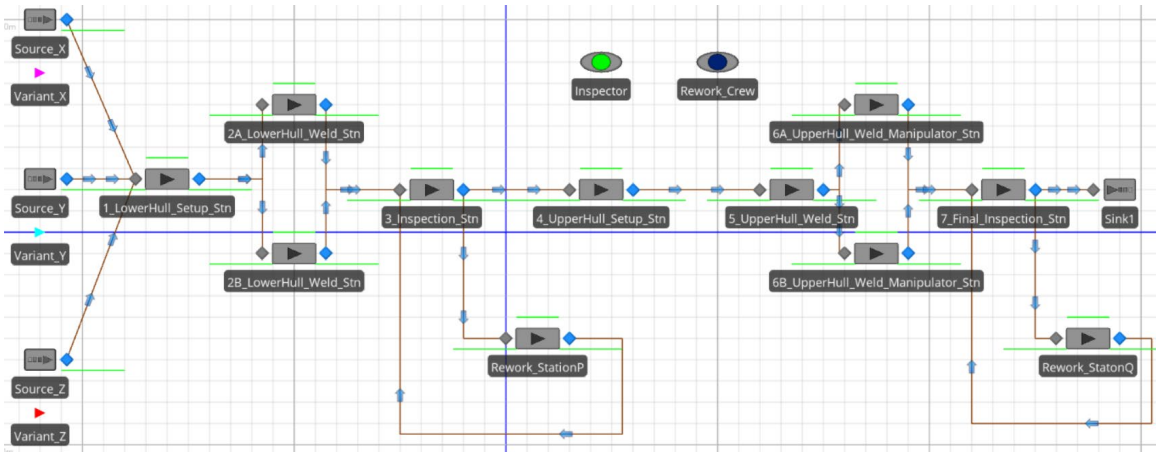

Figure 7. Simio model of the production line

The source object creates entities, which are our three variants (Variant_X, Variant_Y, Variant_Z), that flow through the network to simulate the product flow. These entities exit the system by flowing into the sink node. Every source object has an output node (denoted by the blue node on the right side of the source object) where the model logic can be modified. Additionally, we used three different source objects to produce the three variant entities and set different colors for the three variant entities to identify them. We can observe the product flow visually this way to get an idea of where each product is and where the queue buildups are when the model is running. These entities are tracked by the software to produce the model output parameters such as the number of entities processed at a particular server object.

The seven work stations are represented using server objects. These server objects are named after the main work process that occurs at that station and the respective station number. Every server object has both an input (denoted by the grey node on the left side of the server object) and an output (denoted by the blue node on the right side of the server object). The model logic for the server object can be modified to change the processing time and distribution type for each station which in this case is set to exponential distribution with the respective mean values for each variant as stated in Table 1. We used the exponential distribution as the underlying model assumption that the processes occur independently and continuously at a constant average rate. The capacity for each station indicates the number of WIPs that it can process at any point of time. For our model, the

station capacity represents the station availability. Hence the station capacity value is either one or zero.

The two rework stations are also represented using server objects. The only difference between a rework station and a processing station is that upon completion of the rework, it is fed back into the inspection station. In the event that it fails the quality inspection, it is sent back to the rework station. We set an arbitrary percentage of WIP that will require rework by changing the output logic using link weights.

The sink object is where all the created entities are destroyed, which can be interpreted as the WIP leaving this segment of the production phase. The sink object only has an input (denoted by the grey node on the left of the sink object). It can also be observed that only one sink node is used as compared to the source nodes where we used three. There is no additional benefit to separate the sinks for specific variants as this is just the simulation model's way of letting the entity exit the system.

There are two resource objects in the model, one to represent quality inspectors and another to represent the rework crew. The purpose of the two resource objects is to model the inspector and rework crew as a resource to be used by the inspection and rework stations respectively. This allows the simulation model to create the scenario of the rework crew being occupied at one of the rework stations, while a WIP at another rework station has to wait for the rework crew. This scenario is also applicable for the inspector crew.

For the three scenarios in our study, we vary Station 2A's capacity between one and zero using different modelling techniques to simulate concept drift in the three scenarios so that we can test if the NARX algorithm can detect the change. As Station 2A is the only station with capacity change, it would be easier to test for concept drift detection at one station compared to testing at several stations. A possible extension of this thesis work is to have multiple station capacity reduction in the model. Next, we will discuss how we set the station capacity to change in the different scenarios within the simulation model.

To create the deterministic scenario where the station capacity changes at fixed intervals, we can use the work schedule function in Simio. Figure 8 shows the method of changing the station capacity value at fixed intervals using work schedules. We can set the

capacity of the work station under the "Value" column to change at fixed intervals. The "Cost Multiplier" value can be used if modeling costs are a factor in the study. This method is suitable for shift time intervals that are smaller than a 24-hour period.

Start Time	Duration	End Time	Value	Cost Multiplier	Description
12:00 am	8 hours	8:00 am	1	1	
8:00 am	4 hours	12:00 pm	0	1	
12:00 pm	8 hours	8:00 pm	1	1	
8:00 pm	4 hours	12:00 am	0	1	

Figure 8. Deterministic change of station capacity using work schedules

However, in our scenario the fixed interval durations are 200 hours for the station capacity up time and 80 hours for the station capacity down time respectively. Therefore, changing the station capacity using the work schedule function is not feasible as the durations we are interested in exceeds the 24-hour period. Hence, we take a different approach by using the Add-on processes function in the Simio software. In this function, we can define custom processes that change the model parameters and they trigger when certain conditions are met. Figure 9 shows the process of how Station 2A's capacity changes at fixed intervals.

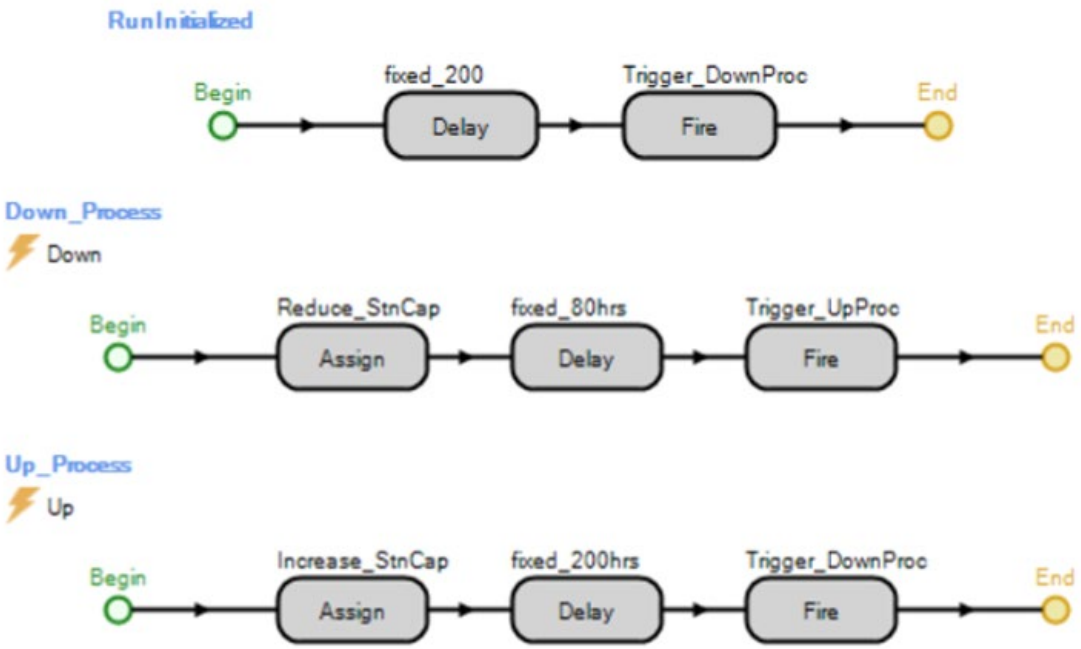

Figure 9. Change of Station 2A's capacity at fixed intervals using the add-on process function

First, we set the initial capacity of Station 2A to one. When the simulation model starts running, we initialize a 200-hour delay before triggering the first time that the station capacity drops to zero. After an 80-hour delay, the station capacity reverts back to one. This process repeats back and forth making the Station 2A's capacity alternate between zero and one until the end of the simulation runtime.

The second scenario is the random scenario where the station capacity changes at a rate based on an exponential distribution with a mean of 200 hours when the capacity is one and an exponential distribution with a mean of 80 hours when the capacity is zero. We used the same method as the previous scenario to build the random scenario in the model and set the initial capacity of Station 2A to one. Figure 10 shows the process of how Station 2A's capacity changes at random intervals.

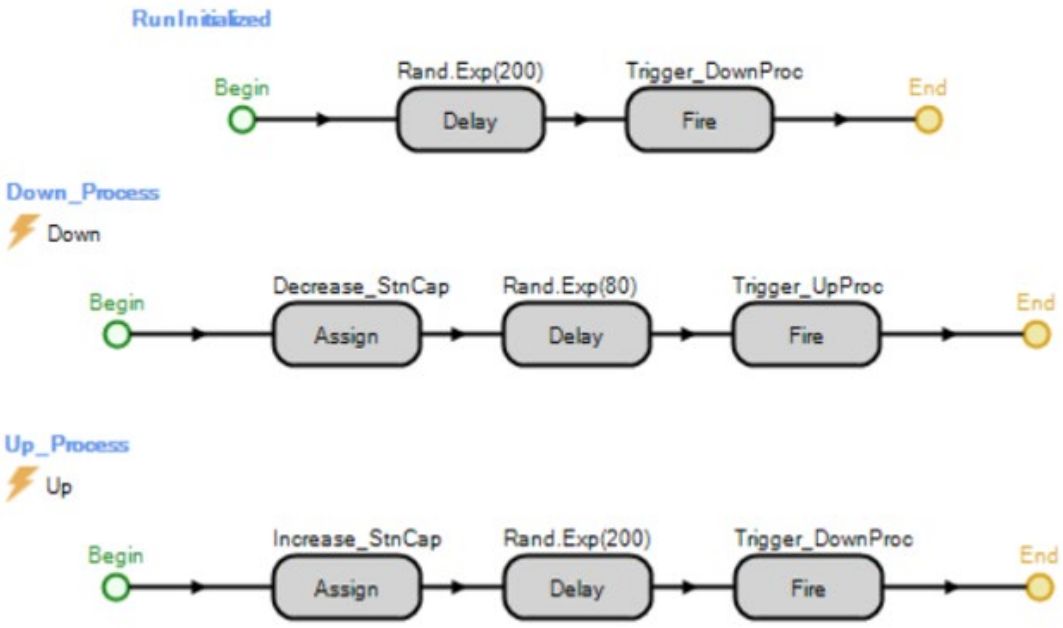

Figure 10. Random change of station capacity using the add-on process function

The processes we define in this scenario is very similar to the ones in the deterministic scenario. The only difference between the two scenarios is the type of delay. For this scenario we use an exponential distribution with mean 200 hours for the station capacity up-time and 80 hours for the station capacity down-time. Similar to the deterministic scenario, Station 2A's capacity alternates between one and zero until the end of the simulation runtime.

The third scenario is the dynamic scenario where the station capacity depends on the real-time state of the system which for our model is the Station 2A's queue. First, we set the initial value of Station 2A's capacity to zero as there is no queue when the model begins. Next, we use the monitor function which allows us to detect threshold crossing state changes. We set two thresholds for the queue length, an upper limit of two and a lower limit of one. When the number of entities in Station 2A's queue is two or more, the first monitor element's add-on process is triggered and the station capacity is set to one. When the number in the queue drops to zero, the second monitor element's add-on process is triggered and the station capacity is set to zero. For the production system this means that

we only require Station 2B to be running and the capacity of Station 2A can be zero when there are few WIPs. But when the amount of WIP increases and the queue length builds up, we will need Station 2A to come back online to prevent bottlenecks. Figure 11 shows the process of how the station capacity changes dynamically.

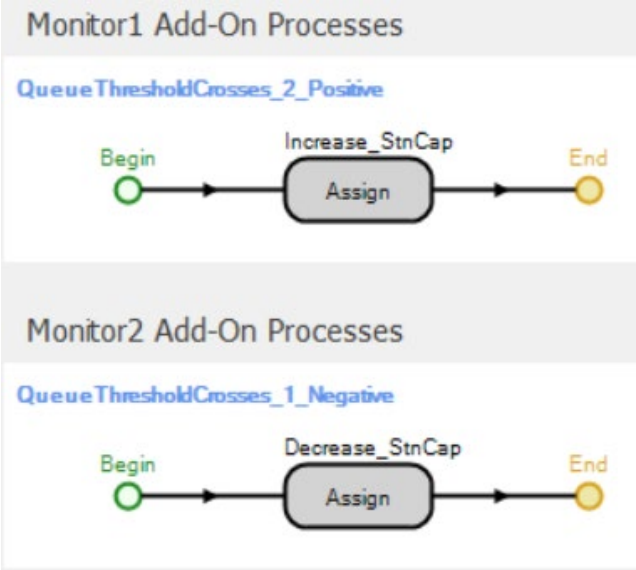

Figure 11. Dynamic change of station capacity using the monitor function in the add-on process

With the model and the three scenarios in place, we can efficiently produce data that reflects the production line status. The experiment function in Simio was used to produce two runs for each scenario to produce the training and validation datasets. Simio uses a pseudorandom number generator and performing the experiment with the same exact parameters will always yield the same result across any computer. Hence it is important that we perform two distinct replications so as to get different outputs for the training and validation dataset. It is worth noting that this is unusual as generally we would run thousands of replications of the simulation model instead of just two. However, the goal of running many replications is to estimate averages which is not our goal. We took the approach of having a long runtime with a warm up period to estimate the steady state behavior of the model and provide long time series as input to the NARX model. We set the warm-up period to be 30 simulation days which is approximately 10% of the total

runtime and a confidence level of 95%. The recorded state variables that reflect the system production state are:

- Queue lengths of each workstation
- Total quantity of each variant in the production system
- Station 2A's capacity

These are many more state variables that can be used such as the station utilization rate and efficiency rate. However, we have picked these three variables as they are objective, definitive, and easily measured in a real-world production line. Subsequently, we export the simulation model output data into an Excel spreadsheet so that we can feed it into MATLAB for the NARX algorithm processing. We accomplish this by having an add-on process for all scenarios that uses a user-defined function called "ExcelWrite" which records all the parameters that we define in this function into specific columns of a designated Microsoft Excel file at specified time intervals. Figure 12 shows the user defined "ExcelWrite" function in the add-on process.

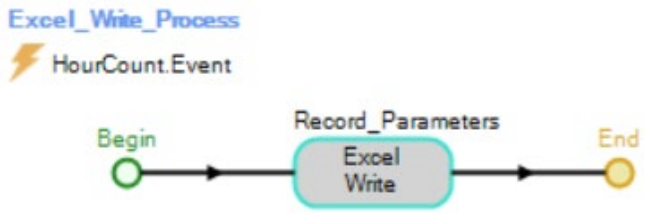

Figure 12. "ExcelWrite" function in the add-on process

Lastly, we will review the several modeling assumptions that were made. The first assumption is that no time is spent transporting the WIPs from station to station. In the real world, the WIPs are moved using overhead cranes and sometimes there can be significant waiting time when the crane is being utilized by other stations resulting in a queue for the overhead crane. The second assumption is that there will always be sufficient manpower for the work stations when they are available. In the real world, the technicians may fall ill

from time to time resulting in reduced manpower availability. The third assumption is that there will always be sufficient space in the production floor for queues to form.

By making these assumptions, we can define the boundary of our model and focus on how the changes in station capacity affect the queue lengths of the other stations in the system. It is important to define model boundaries as it is impossible to include every element and consideration in the model (Bellinger & Fortmann-Roe, 2017). Having a problem boundary also helps us separate the highly relevant features from the less relevant ones.

C. NARX ALGORITHM WITH SCALED DOWN MODEL

With the simulation model for the production line ready, we now move on to the NARX algorithm. The NARX algorithm code consists of three main segments: the data input, NARX parameter setting, training the model, and model validation. Now we will discuss what each segment's role is in the algorithm.

The first segment of code deals with data input where we feed the output data in the Excel spreadsheet obtained from the simulation model. There is an additional processing step of converting the simulation output data with the "con2seq" MATLAB command. This command converts the raw data into cell inputs which enables the data to be used in the NARX algorithm. As there is no limit to the number of simulations we can conduct, technically we have infinite amount of data that we can input into the NARX algorithm. Therefore, we do not need to split the data into training and validation sets. Instead, we will use the output of two independent runs for the training and validation data sets. We then create two sets of input, the primary time series and the secondary time series for the training dataset and another two sets for the validation dataset.

In the second segment which sets the NARX parameters, we define the time lags for the primary time series, secondary time series, and the number of hidden neuron layers in the network. To meet the two goals of our thesis which is to predict the station capacity and queue length, we explore two time series with different variable combinations for the primary and secondary time series as shown in Table 2. Next, we test a range of values for the number of hidden neuron layers and the number of time lags for the primary and

secondary series. For our case, we will set an arbitrary range of 1-10 to test for the time lags and 1-20 for the number of hidden neurons. Note that the range of NARX parameter values to test can vary for different case studies.

Table 2. Time series with different variable combinations for the primary and secondary time series

No.	Primary Time Series	Secondary Time Series
1	Station 2A Capacity	• Queue lengths of all workstations • Number of WIP of each variant
2	Station 2A Queue Length	• Queue lengths of all workstations except Station 2A • Number of WIP of each variant • Station 2A's Capacity

Instead of testing every combination for the three parameters (time lag for primary time series, time lag for secondary time series, and number of hidden neuron layers), we use a NOLH sampling technique to reduce the number of parameter combinations to test for while still getting a good representation of the dataset. We review this in greater detail in Chapter III Section D. Once the ideal parameter values are determined, we can proceed with training and validating the model.

In the final segment of code, we train the model using the first set of data of the primary and secondary time series. The first set of data forms the training dataset and we prepare it by using the "preparets" function in MATLAB. This function shifts the primary and secondary time series by the required number of steps to fill the initial input and layer delay states. Figure 13 shows the NARX network that is produced by the algorithm based on our parameter input of three time lags for the primary time series, three time lags for the secondary time series and three neuron layers where y(t) is the primary time series and x(t) is the secondary time series. Next, we use the second set of independent data to validate the trained model. Therefore, the fitted network on the training data is applied to the validation data for prediction. The output generated from this segment is the one step ahead performance and MSE value from the validation dataset. Additionally, we also can plot a

graph comparing the predicted values of the model versus the actual data for visual observations.

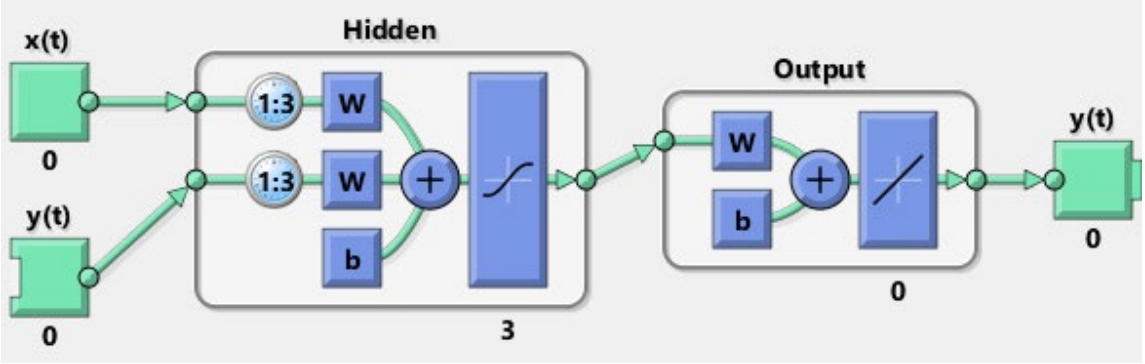

Figure 13. Sample NARX network generated using MATLAB

Finally, we rerun the algorithm with the datasets from the three scenarios and change the primary time series from the station capacity to the queue length for each of them. To change the primary time series, we replace the existing column that is being read into the algorithm to the one that we wish to test for. In total, we have three Simio models illustrating the three scenarios (deterministic, random, dynamic) and six MATLAB NARX algorithms (three sets of code for each scenario where the primary time series is the Station 2A capacity, and another three set of code where the primary time series is the queue length of Station 2A).

D. PARAMETER TUNING

In the second segment of the NARX algorithm code, we stated that the NARX parameters selection process can be streamlined using NOLH to reduce the number of combinations to test for. This is important as testing all the possibilities would require us to perform over 2000 runs for each scenario and alternate primary time series. Hence, we implement a NOLH to reduce the number of runs required while still being able to obtain quantitative parameter values from multiple dimensions. We use a NOLH Excel spreadsheet to perform this task (Sanchez, 2011). The objective is to determine which combination of values for the parameters will yield low MSE values. First, we input the

range of values that each value can take which for our scenario is 1-10 for both the primary time series y(t) time lags and the secondary time series x(t) time lags as well as 1-20 for the number of hidden neuron layers. Next, we generate the input neural network values to test for each parameter by entering the respective range of values into the NOLH Excel spreadsheet. Finally, we input the parameter combinations into the NARX algorithm to generate the training data MSE as well as the validation data MSE.

Now we will briefly describe what the number of x(t) and y(t) time lags for the primary and secondary time series as well as the number of hidden neuron layers mean. The number of x(t) and y(t) time lags determine how many time steps prior are being considered by the NARX algorithm when evaluating the current time step to make a prediction. The larger the number of time lags, the further the algorithm looks back. While having a larger time lag gives the NARX algorithm more data to work with and may produce more accurate predictions, it may not be necessary as the values may have positive dependence where the next value will be closer to the current value compared to several values prior.

To reduce complexity and the dataset size, we use a scaled down model of the production system. Figure 14 shows the scaled down simulation model that we use to generate data for feeding into the algorithm to fine tune the NARX parameters. The scaled down model consists of three stations and three variant sources. We used notional values for the interarrival time and station processing time in the scaled down model for the purposes of determining the NARX input parameters. The exact values do not need to match the real-world production values, but the chosen values must ensure queue stability where the traffic intensity per server does not exceed one.

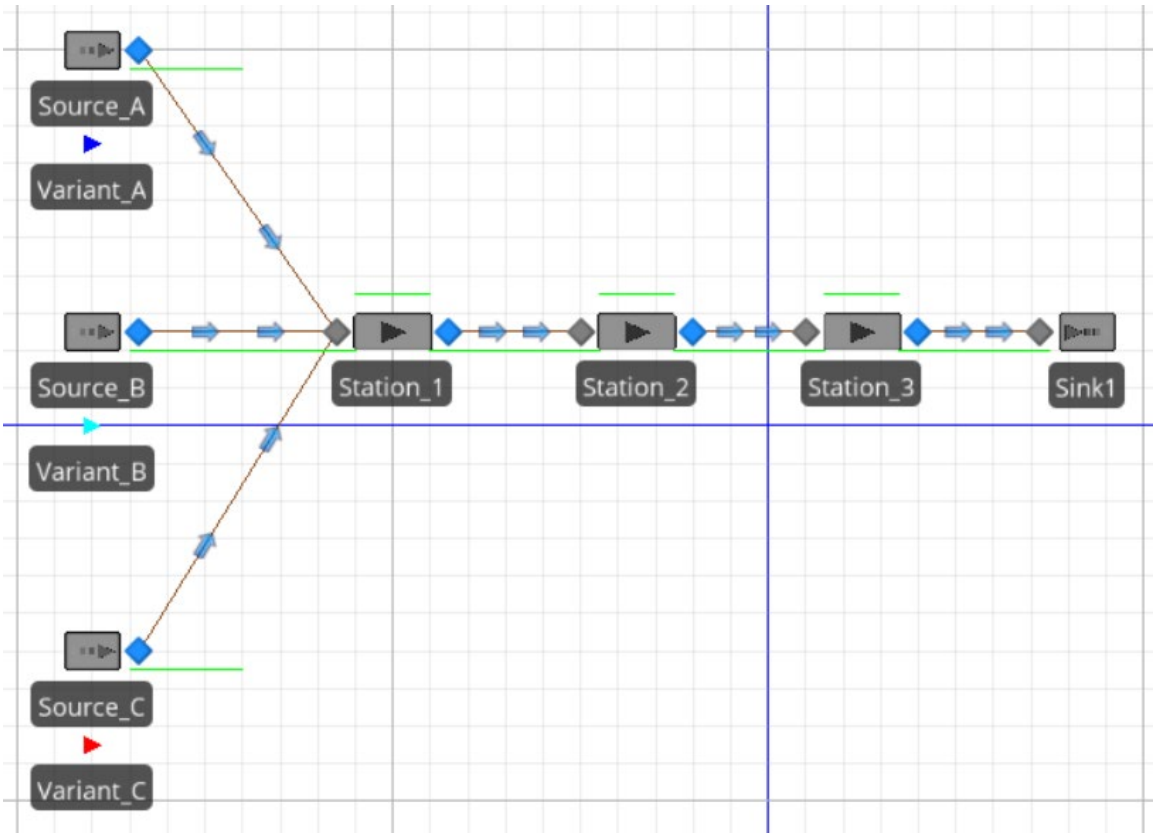

Figure 14. Scaled down simulation model

Using the NOLH spreadsheet, we run 17 different combinations of the number of *x(t)* and *y(t)* time lags and number of hidden neuron layer parameters. Table 3 shows the parameter combinations and respective MSE values for the deterministic scenario. While collecting the data, we noticed that the MSE values will fluctuate with each run. However, the variation between values is small when we rerun the NARX algorithm with the same set of parameters. The variation in the results can be due to the nature of the algorithm (Brownlee, 2020). This is not an issue for us as the goal is to access the overall quality of results produced by the NARX algorithm rather than computing specific MSE values.

Table 3. NARX input parameter NOLH combinations and MSE values

x(t)	*y(t)*	No. of neuron layers	Training MSE	Validation MSE
4	10	16	0.0286	0.0423
2	3	18	0.0314	0.0483
2	5	2	0.0327	0.0423
3	7	7	0.0320	0.0463
8	9	9	0.0275	0.0417
10	4	8	0.0285	0.0394
7	3	20	0.0273	0.0475
6	9	15	0.0314	0.0478
6	6	11	0.0274	0.0433
7	1	5	0.0301	0.0395
9	8	3	0.0293	0.0380
9	6	19	0.0268	0.0448
8	4	14	0.0279	0.0376
3	2	12	0.0302	0.0476
1	7	13	0.0319	0.0479
4	8	1	0.0333	0.0403
5	2	6	0.0278	0.0386

For our study, we determine that there is no meaningful difference in the MSE values across the different combinations with respect to the overall impact on the model. With this in mind, we should try to use a smaller value for the parameters to reduce the size of the network. Hence, we repeat the process with a smaller range of 1-3 for the three parameters to produce a smaller sized network. As the total number of combinations is 27 and relatively small, we can afford to do a full factorial design which tests every possible combination of the parameters. Table 4 shows the full factorial design for the parameters with values ranging between 1-3.

Table 4. Full factorial design of NARX input parameter for the range 1-3

x(t)	*y(t)*	No. of neuron layers	Training MSE	Validation MSE
1	1	1	0.0339	0.0406
1	1	2	0.0339	0.0407
1	2	2	0.0337	0.0410
1	2	1	0.0341	0.0408
1	2	3	0.0339	0.0410
1	3	3	0.0339	0.0408
1	3	1	0.0339	0.0407
1	1	3	0.0338	0.0406
1	3	2	0.0339	0.0404
2	1	1	0.0338	0.0405
2	1	2	0.0340	0.0445
2	2	1	0.0338	0.0408
2	2	2	0.0330	0.0410
2	3	3	0.0354	0.0464
2	2	3	0.0327	0.0403
2	1	3	0.0331	0.0409
2	3	1	0.0339	0.0407
2	3	2	0.0323	0.0413
3	3	2	0.0318	0.0451
3	1	3	0.0326	0.0400
3	3	3	0.0316	0.0397
3	1	2	0.0319	0.0422
3	3	1	0.0337	0.0406
3	2	1	0.0338	0.0404
3	2	3	0.0332	0.0398
3	2	2	0.0338	0.0404
3	1	1	0.0337	0.0405

From the full factorial design, we can observe that the variance between MSE values across the different combinations is small (3 decimal places) and the combination of (3, 3, 3) which represents (lag of *x(t)*, lag of *y(t)*, number of hidden neuron layers), yields the lowest training and validation MSE of 0.0316 and 0.0397 respectively. Compared to the combination (8, 4, 14) which has the lowest validation MSE value of 0.0376 in Table

3, the difference is approximately 10% which is small enough for us to use the smaller network model. Therefore, we will use the combination of (3, 3, 3) which is three time lags for the primary time series, three time lags for the secondary time series, and three hidden neurons in the layer.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS AND ANALYSIS

Chapter IV discusses the results obtained from the NARX algorithm for the various scenarios and response variable combinations. The naming convention used to describe the scenario and response variable combination will be as follows: "[scenario name]-[response variable]", where the "scenario name" describes one of the three scenarios (deterministic, random, dynamic), and "response variable" describes one of the two responses used as the primary time series (queue, station capacity). For the first set of results, we evaluate the validation MSE at different time steps. Subsequently, we explore some results with alternative inputs for the NARX algorithm. The first alternative experiment is where we train the NARX algorithm on one type of scenario and check the prediction accuracy for another type of scenario. In the second alternative experiment, we reduce the number of input variables for the secondary time series and review the corresponding model validation MSE values.

A. MSE ACROSS DIFFERENT TIME STEPS

Determining how frequently data is being collected (such as the queue length for each station and the quantity of WIPs in the production system) can have real world implications. By having a lower data collection frequency, we can potentially have cost savings on manpower that is allocated to collect data. However, collecting data more frequently can lead to better predictions. Hence, we evaluate the NARX algorithm's performance across different interval time steps. We generated data at different interval time steps (one hour, four hours, and eight hours) and input it into the NARX algorithm to produce the validation MSE values for the various scenario-response variable combinations. The MSE values for these three time step intervals are computed across the entire runtime and are called the fixed time steps. We also compute another set of MSE values at discrete event change times by creating a function in MATLAB to compute the MSE only when the true value of the predicted response variable changes. We call this the variable time step. For example: when trying to predict the station's queue length, we only consider times when the number in the queue changes and the true value is compared with

the prediction to calculate the MSE value. This would give us a total of four different time steps of MSE values which are the three fixed time step intervals of one hour, four hours, eight hours, and the variable time step for the various scenario-response combinations. We developed the additional variable time step computation as the time step interval chosen affects the MSE values as seen in Table 5. The smaller the time step, the higher the prediction accuracy because the system state is not changing as much in the short term. Hence, it is important to develop the variable time step case to check specifically how the prediction performs at the discrete event change times using the variable time step.

It should be noted that the MSE values generated from the NARX algorithm also fluctuates similarly to what we experienced while performing the parameter tuning with the NOLH. As mentioned in Chapter III Section D, the MSE values do not deviate much and have the same order of magnitude when we rerun the NARX algorithm with the same set of parameters and datasets. The validation MSE values for the scenarios are shown in Table 5. We will now review the results for each time interval.

Table 5. Validation MSE at different time steps

Scenario – Response	**Interval Time Step**			
	1-Hour Time Step	**4-Hour Time Step**	**8-Hour Time Step**	**Variable Time Step**
Deterministic – Queue	0.0382	0.1674	0.6105	0.9245
Deterministic – StationCapacity	0.0075	0.0278	0.0541	0.9970
Random – Queue	0.1010	0.4334	0.8444	2.3196
Random – StationCapacity	0.0078	0.0385	0.0832	0.9613
Dynamic – Queue	0.0269	0.1266	0.1773	2.0965
Dynamic – StationCapacity	0.0260	0.0157	0.0293	0.5992

1. Sampling at a 1-Hour Time Step

We can expect the MSE values for the one-hour time step intervals to be the lowest among all of the different time steps. This is because when we sample in small time intervals, the system is not changing much with each time step. Therefore, it is easier for the algorithm to predict that the system will stay the same. Reviewing the MSE values, the Deterministic-Queue MSE value was 0.0382 which is one order of magnitude bigger than

the Deterministic-StationCapacity MSE of 0.0075. This could be attributed to the NARX algorithm having an easier time to predict between zero and one for the station capacity whereas the values for the queue have a larger range between zero and nine based on the modeling parameters we have set. Figure 15 shows the plot for the Deterministic-StationCapacity while Figure 16 shows the plot for the Deterministic-Queue. The blue lines are the true values and the red lines are the predicted values by the NARX algorithm. As the MSE values are small, it is difficult to see the difference visually on the plot.

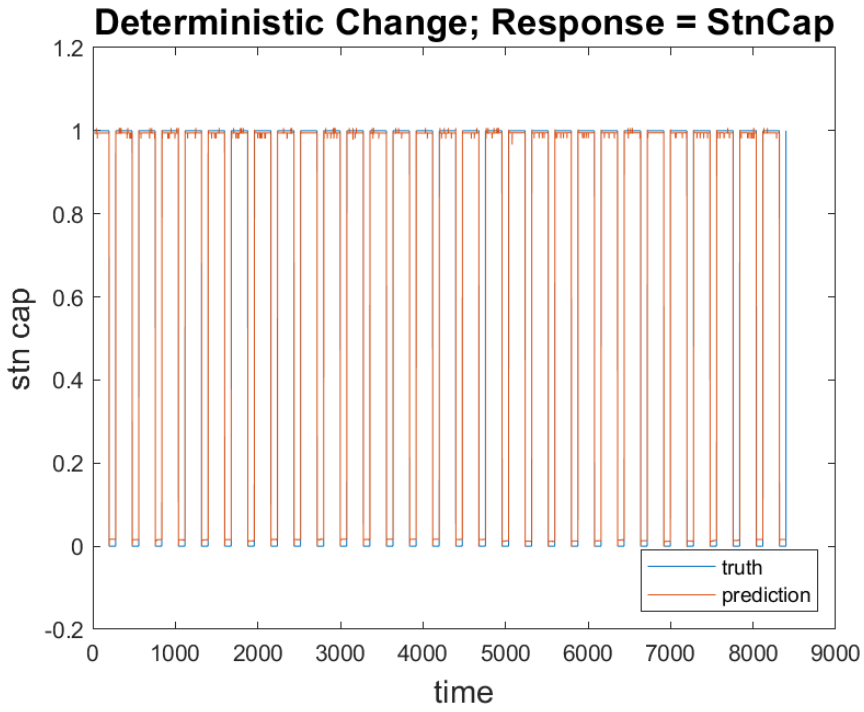

Figure 15. Plot for Deterministic-StationCapacity, 1-hour time step

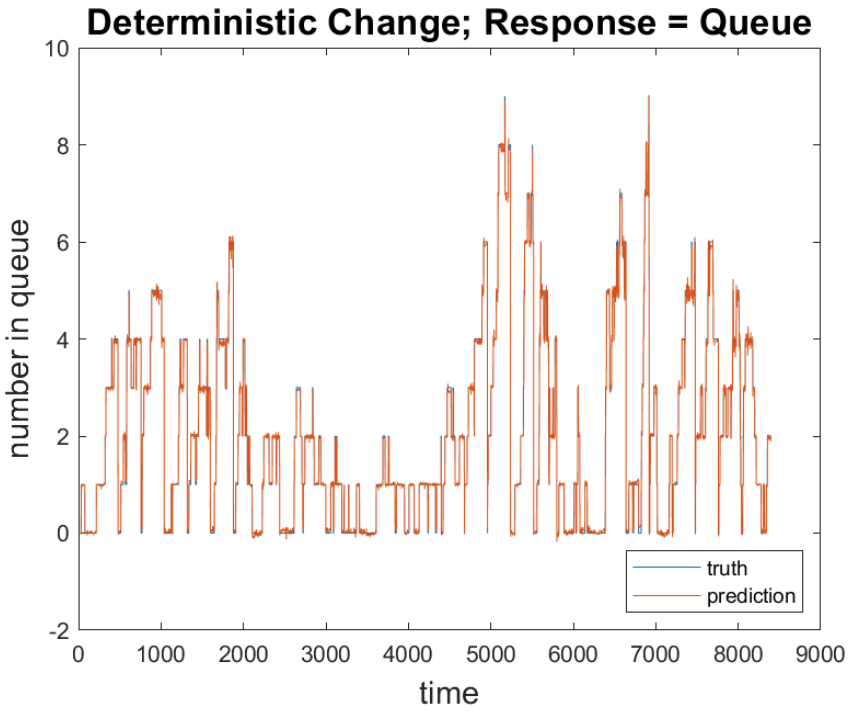

Figure 16. Plot for Deterministic-Queue, 1-hour time step

For the random scenario, the Random-StationCapacity MSE value of 0.0078 is smaller by an order of magnitude of two compared to the Random-Queue MSE value of 0.1010. The difference in the order of magnitude may be considered as one if rounding is taken into account. Interestingly, the Random-StationCapacity had a similar value to that of the Deterministic-StationCapacity even though the distribution type for the station capacity of the two scenarios was different.

Lastly, there was no major difference between the Dynamic-Queue MSE value (0.0269) and Dynamic-StationCapacity MSE value (0.0260). This would suggest that the NARX algorithm performs equally well for both responses in the dynamic scenario when the time step is small. We also observed that the MSE value for the Dynamic-StationCapacity was one order of magnitude larger than the other two scenarios when StationCapacity was set as the response.

Comparing across all the MSE values for the one-hour time step interval, the Random-Queue has the highest MSE value. This is expected as the random distribution is

more difficult to predict than the other two scenarios and the queue variable can take on a larger range of values compared to the station capacity variable.

2. Sampling at a 4-Hour Time Step

The MSE values for the four-hour time step intervals were larger than those in the one-hour time step intervals as seen in Table 5. Comparing the MSE values for the deterministic scenario at the 4-hour time step, the Deterministic-Queue MSE had a value of 0.1674 which is one order of magnitude larger than the Deterministic-StationCapacity value of 0.0278. Figure 17 shows the Deterministic-Queue plot for the four-hour time step interval. Through visual comparison between the Deterministic-Queue plot for the four-hour time step and one-hour time step, we can see the difference between the truth and prediction values more distinctly compared to the one-hour time step.

The Random-Queue MSE value of 0.4334 is larger by an order of magnitude compared to the Random-StationCapacity MSE value of 0.0385. The Random-Queue MSE value was also the largest value when comparing across all the MSEs in this time step interval.

The Dynamic-Queue MSE value of 0.1266 was one order of magnitude larger than the Dynamic-StationCapacity value of 0.0157. This is one of the differences we observed when the time step was changed from one-hour to four hours as there was no difference in the order of magnitude between the Dynamic-Queue and Dynamic-StationCapacity for the one-hour interval, thus it is more difficult to predict over a larger time step. Additionally, we also observed that the Dynamic-StationCapacity MSE value decreased when the time interval was set to four hours compared to the one-hour time step interval. However, this is not a cause for concern as the order of magnitude between the Dynamic-Station Capacity MSE values for the one-hour and four-hour time step is the same and the difference is small (less than 0.01). We hypothesize that this is due to the variation resulting from the nature of the NARX algorithm training. We also hypothesize that the algorithm is able to produce similar MSE values for the one-hour and four-hour time steps because the dynamic changes occur in time steps larger than four hours. Lastly, we observed that all the MSE values for

the StationCapacity response had the same order of magnitude for all three scenarios (0.0278, 0.0385, 0.0157).

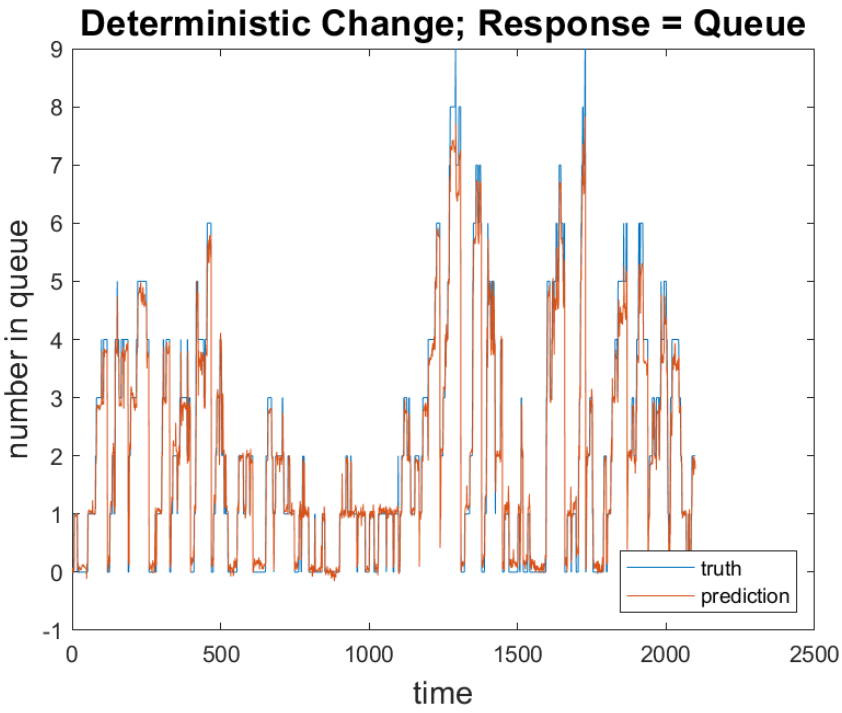

Figure 17. Plot for Deterministic-Queue, 4-hour time step

3. Sampling at an 8-Hour Time Step

The MSE values for the eight-hour time step intervals were larger than those in the one-hour and four-hour time step intervals as seen in Table 5. Similar to the previous time steps, the Deterministic-Queue MSE value of 0.6105 was one order of magnitude larger than the Deterministic-StationCapacity MSE value of 0.0541. Figure 18 shows the Deterministic-Queue plot for the eight-hour time step interval. We can see the difference between the truth and prediction values even more distinctly as the time step becomes larger because it more difficult to predict state changes over longer time periods.

The Random-Queue MSE value of 0.8444 is larger by an order of magnitude compared to the Random-StationCapacity MSE value of 0.0832. The Random-Queue MSE value was also the largest value when comparing across all the MSEs in the eight-hour time

step interval. The Dynamic-Queue MSE value of 0.1773 was one order of magnitude larger than the Dynamic-StationCapacity value of 0.0293.

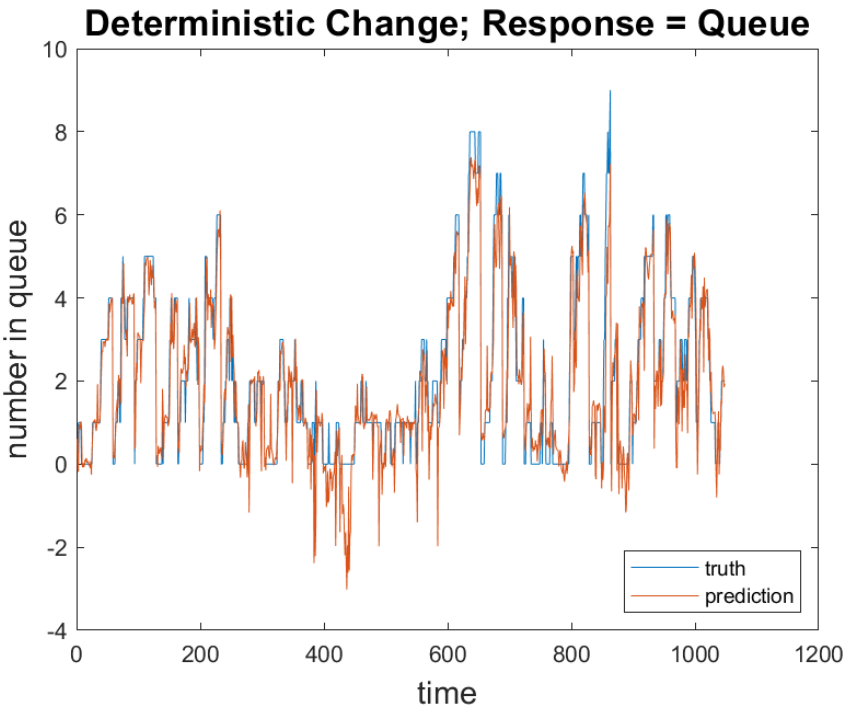

Figure 18. Plot for Deterministic-Queue, 8-hour time step

Lastly, we can observe from Figure 18 that there are some instances of the prediction having a negative value for the number in queue which is not possible in the real-world. For example, after time 400 the prediction value goes below zero for a period of time. Hence, a possible improvement to the algorithm could be to change negative values to zero by adding a constraint.

4. Sampling Using a Variable Time Step

The MSE values generated by the NARX algorithm for the variable time step were larger compared to those from the one, four, and eight-hour time step interval for all the scenario-response tests. The larger validation MSE values suggest that the algorithm is good at predicting the smaller time intervals when the system is less likely to have changed, which is taken into account in the previous three fixed time step intervals. We observe that

the Random-Queue and Dynamic-Queue MSE values of 2.3196 and 2.0965 respectively are one order of magnitude larger than the Deterministic-Queue MSE value of 0.9245. We can hypothesize from this that the NARX algorithm does not perform as well when the response variable is set to queue and the scenario is set to random or dynamic.

Lastly, the MSE values for the StationCapacity response had the same order of magnitude for all three scenarios. The MSE values are as follow: Deterministic-StationCapacity (0.9970), Random-StationCapacity (0.9613), Dynamic-StationCapacity MSE value (0.5992).

B. DIFFERENT SCENARIO DATASET FOR TRAINING AND VALIDATION DATASET

We try to draw some insights on the NARX algorithm by using different datasets to train and validate the model. The idea behind this approach is to check if the NARX algorithm's prediction is overfitting by comparing the prediction against another time series that it was not meant to predict. Table 6 shows the training and validation MSE values when we train the NARX algorithm on one type of scenario and how well it predicts for another type of scenario. These values were generated based on the one-hour interval setting which generally had very low MSE values. The naming convention used to describe the response variable, training and validation combination will be as follows: [response variable]-[training scenario]-[validation scenario]", where the "scenario name" describes one of the three scenarios (deterministic, random, dynamic), and "response variable" describes one of the two responses (queue, station capacity).

Table 6. Training and validation MSE values for different combinations of training and validation datasets (1-hour time step interval)

Response Variable	Training Scenario Dataset	Validation Scenario Dataset	Training MSE	Validation MSE
Queue	Deterministic	Random	0.0358	0.1201
	Deterministic	Dynamic	0.0305	0.0270
	Random	Deterministic	0.0446	0.0888
	Random	Dynamic	0.0447	0.0270
	Dynamic	Deterministic	0.0266	0.1062
	Dynamic	Random	0.0265	0.2201
StationCapacity	Deterministic	Random	0.0071	0.0156
	Deterministic	Dynamic	0.0071	0.0072
	Random	Deterministic	0.0069	0.0073
	Random	Dynamic	0.0066	0.0193
	Dynamic	Deterministic	0.0034	0.1696
	Dynamic	Random	0.0033	0.3074

We split the results based on the response variable, Queue and StationCapacity, and the different combinations of training and validation datasets. We compared the validation MSE values in Table 6 against the ones in Table 5, where the same dataset was used for training and validation, and observe that there are three categories which the various combinations fall under for this result experiment. The combinations in the first category are the ones that perform as expected, whereby the validation MSE value is larger than the training MSE value and has an equal or larger order of magnitude. For example, the combination of Queue-Deterministic-Random had a training MSE value of 0.0358 and validation MSE value of 0.1201. We observe that 8 out of 12 of these combinations fall within the first category. The second category consists of combinations that did not yield much difference between the training and validation MSE values as well as the order of magnitude. For example, the combination of StationCapacity-Random-Deterministic had a training MSE value of 0.0069 and validation MSE value of 0.0073. We can also observe that most of the combinations which fall in the second category had station capacity as the response variable. As we had previously hypothesized, this is likely due to the small range of values (zero and one) that the station capacity fluctuates between which enabled the NARX algorithm to make relatively accurate predictions. We observe that 2 out of 12 of these combinations fall within the second category. Lastly, the third category is where the

validation MSE value is lower than the training MSE value. For example, the combination of Queue-Random-Dynamic had a training MSE value of 0.0447 and validation MSE value of 0.0270. We observe that 2 out of 12 of these combinations fall within the third category. Although the MSE values have the same magnitude, this is a possible unusual occurrence as the results are suggesting that the model is performing better on the test rather than what it was trained for. We hypothesize that there could be possible stochastic sampling error, sampling bias, and variability in the MSE across multiple computations. In order to confirm this, additional work will need to be done by performing multiple runs on the algorithm and determining the confidence intervals of the mean.

C. REDUCED INPUT VARIABLES

The concept of using fewer input variables can have real world implications as requiring fewer inputs will mean having to collect less data on the production floor which can be a cost saving. We should also take into account that there might be some variables that have minimal effect on the response variable, therefore a few input variables might also be enough to produce an accurate prediction. Hence, we reduce the number of input variables that are fed into the NARX algorithm and evaluate its performance across several combinations. As there are many variable combinations in each scenario to test for, we will focus on a few combinations in the one-hour time step interval for the deterministic scenario. The training and validation MSE for the combinations we tested for can be found in Table 7. The naming convention used to describe the input variable and response variable combination will be as follows: [input variable]-[response variable], where the "input variable" describes the data that is used to train the model and the "response variable" describes the data used as the response to be predicted. The term "all variables" represents the station capacity, queue length for the different stations, and quantity of each variant in the production system.

Table 7. Training and validation MSE values for several combinations of input and response variables (Deterministic Scenario 1-hour time step interval)

Row No.	Input variable	Response variable	Training MSE	Validation MSE
1	all variables	2A queue length	0.0302	0.0382
2	2A capacity	2A queue length	0.0307	0.0366
3	all variables	2A capacity	0.0070	0.0075
4	2A queue length	2A capacity	0.0071	0.0076
5	2B queue length	2A queue length	0.0544	0.0872

We observe that the "all variables-2A queue length" combination generated training and validation MSE values had the values of 0.0302 and 0.0382 (row 1). These values are similar to "2A capacity-2A queue length", which have training and validation MSE values of 0.0307 and 0.0366 (row 2) which also is of the same order of magnitude. A similar observation is also made for "all variables-2A capacity" training and validation MSE values of 0.0070 and 0.0075 (row 3) and 2A queue length-2A capacity training and validation MSE values of 0.0071 and 0.0076 (row 4) whereby the values are similar and have the same order of magnitude. From these results, we can hypothesize that there is potential to reduce the number of input variables while still achieving a MSE value that is similar to the one if all variables are used.

Lastly, we tried an additional combination of "2B queue length-2A queue length" which produced the training and validation MSE values of 0.0544 and 0.0872 respectively (row 5). While the order of magnitude remained the same when comparing against the combinations where the response variable is set to Station 2A's queue length, the MSE value increased. We hypothesize that the NARX algorithm is able to perform relatively well even with a reduced number of input variables as long as there is some relationship with between the variables chosen for training and prediction. Beyond the scope of evaluating the NARX algorithm, possible additional work can be done in this area by performing regression analysis to determine the relationship of the input and response variables.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION AND FUTURE WORK

In this chapter, we summarize the results obtained from this study. We also present some possible ideas for future work by expanding on various areas of the NARX algorithm and simulation model.

A. CONCLUSION

Concept drift is a significant change in system behavior that may cause a model to be outdated due to changing conditions. In an armored vehicle manufacturing setting, a resource failure (decrease in station capacity) can lead to queue buildup. This can lead to greater problems such as delayed production and floor space constraints. By using the NARX algorithm to predict the future state of a production system given some existing data, we try to detect concept drift. By being able to detect changes as soon as they happen, we can potentially generate significant cost savings by adding resources to overcome bottlenecks in the system and prevent late delivery penalties. There can also be broader implications in this study in terms of developing methods for detecting key changes or unusual activities in the system and mapping them to events of interest.

To test the NARX algorithm's performance in accurately predicting future values, we developed a simulation model with three scenarios where the station capacity fluctuates differently (deterministically, randomly, and dynamically) using Simio to generate input data for the NARX algorithm which was run using MATLAB. The NARX algorithm requires three key parameter inputs which are the time lags for the primary time series (variable of interest), time lags for the secondary time series (exogenous variables), and the number of hidden neuron layers. Depending on the range of values we wish to test for these parameters, the number of runs required to determine the optimal values can be very large. Hence, we employed an NOLH to determine these parameters and found that having three time lags for the primary and secondary time series as well as three hidden neurons in the layer would work well in our setting.

After we had tuned the NARX algorithm parameters, we evaluated the model's performance in prediction accuracy based on the validation MSE values in Chapter IV. We

also explored alternative experiments to study how the NARX algorithm would perform under different scenarios.

The general takeaways are that the NARX algorithm produces more accurate predictions when the time step is small and when there are fewer change-times for the variable of interest. The random scenario-queue response variable combination had the highest MSE values across the different time steps. We hypothesize that the NARX algorithm does not produce predictions for the random scenario that are as accurate as the ones from the deterministic and dynamic scenarios because random system changes are harder to predict. Another possible reason for this combination having the highest MSE value could be due to the queue variable being able to take on a larger range of values compared to the station capacity variable. For the comparison of the NARX algorithm predictions against other time series that it was not trained for, we observed that there were three categories of results. The first category are the ones that perform as expected where the validation data had a larger value and order of magnitude compared to the training data. The second category are the combinations that had small differences between the training and validation MSE values. This would suggest that the NARX algorithm is able to predict accurately for those scenario-response combinations. The third category are the combinations that had a lower validation MSE than the training MSE. We hypothesize that this is possibly due to stochastic sampling error, sampling bias, and variability in the MSE across multiple computations. Lastly in the reduced input variables experiment, we tested a few combinations using subsets of the input variables and the NARX algorithm was able to produce results similar to those when using all the available variables. Hence, we can hypothesize that the NARX algorithm can perform well even with reduced inputs as long as there is a relationship between the variables that the algorithm is able to identify.

B. FUTURE WORK

The possible future work for this thesis can be split into two categories, expanding the simulation model and improving the NARX algorithm. For the first category of future work on the simulation model, we can explore collecting actual data from a production line and incorporating more real-world features into the model. An example of a real-world

feature that can be modeled is manpower scheduling. In our model, we made an assumption that there will always be sufficient manpower for the work stations. In the real world, manpower scheduling for production systems is a challenging task. It involves the identification of an equitable allocation of working hours for the production crew while trying to increase production efficiency, maximize resource capacity, reduce costs, and ultimately increase profitability of the organization (Shahnazari-Shahrezaei et al., 2013). By adding this aspect to the model, we can provide insight to aid in the manpower scheduling decision making process.

A second category of future work involves improving the NARX algorithm's capabilities. We can explore incorporating additional information, such as seasonal behavior and trends, to help with drift detection. We can also generate data from different types of simulation models to further assess the NARX algorithm capabilities. Lastly, we can consider the ensemble model approach, which combines multiple machine learning algorithms to make the training and prediction values more robust.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Army Technology. (2021, July 22). Bionix infantry fighting vehicle range. https://www.army-technology.com/projects/bionix/

Bellinger, G., & Fortmann-Roe, S. (2017, November 7). The process of modeling: model boundaries [systems thinking & modeling series]. RealKM. https://realkm.com/2017/11/07/the-process-of-modeling-model-boundaries-systems-thinking-modelling-series/

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., & Jenssen, R. (2018, July 20). An overview and comparative analysis of recurrent neural networks for short term load forecasting. arXiv.org. https://arxiv.org/abs/1705.04378

Brownlee, J. (2020, August 26). Why do I get different results each time in machine learning? Machine Learning Mastery. https://machinelearningmastery.com/different-results-each-time-in-machine-learning/.

Defense Acquisition University. (2012, August). Defense manufacturing management guide for program managers (PQM for PMs). DAU Home. https://www.dau.edu/tools/t/Defense-Manufacturing-Management-Guide-for-Program-Managers-(PQM-for-PMs)

Dun & Bradstreet First Research. (2021, June 28). Armored military vehicle manufacturing industry profile. https://www.firstresearch.com/Industry-Research/Armored-Military-Vehicle-Manufacturing.html

Eccles, R. G., Newquist, S. C., & Schatz, R. (2007, February). Reputation and its risks. Harvard business review risk management

Ghalehkhondabi, I., & Suer, G. (2018). Production line performance analysis within a MTS/MTO manufacturing framework: A queueing theory approach. https://doi.org/10.1590/0103-6513.20180024

Ghodous, P., Dieng-Kuntz, R., & Loureiro, G. (Eds.). (2006). Leading the web in concurrent engineering: next generation concurrent engineering. https://ebookcentral.proquest.com

Government Accountability Office. (2021). Defense budget: Opportunities exist to improve DOD's management of defense spending (GAO-21-415T).

Govil, M. K., & Fu, M. C. (1999). Queueing theory in manufacturing: A survey. *Journal of Manufacturing System*s, 18(3), 214–240. https://doi.org/10.1016/s0278-6125(99)80033-8

Hernandez, A. S. (2008). Breaking barriers to design dimensions in nearly orthogonal Latin hypercubes. [Doctoral dissertation. Naval Postgraduate School]. NPS Archive: Calhoun. https://calhoun.nps.edu/handle/10945/38211

Hernandez, A. S., Lucas, T. W., & Carlyle, M. (2012). Constructing nearly orthogonal Latin hypercubes for any nonsaturated run-variable combination. NPS Archive: Calhoun. https://calhoun.nps.edu/handle/10945/38211

Habchi, G. (2003). Modelling and simulation: Analysis, design and optimization of industrial systems [MOSIM'01]. *Simulation Modelling Practice and Theory,* 11(1), 1–3. https://doi.org/10.1016/s1569-190x(03)00039-x

Hallum, T. (2021). Technician performing welding work. *Mad Scientist Laboratory.* https://madsciblog.tradoc.army.mil/311-achieving-an-ai-era-workforce-by-2025-a-modern-scalable-approach-to-retooling-the-united-states-and-its-army/

IBISWorld. (2021, March 25). *Industry market research, reports, and statistics*. https://www.ibisworld.com/united-states/market-research-reports/tank-armored-vehicle-manufacturing-industry/

Janes. (2021, February 17). *Singapore announces budget increase for 2021.* https://www.janes.com/defence-news/news-detail/singapore-announces-budget-increase-for-2021_15567

Joines, J. A., & Roberts, S. D. (1998). Fundamentals of object-oriented simulation. 1998 winter simulation conference. *Proceedings* (Cat. No.98CH36274). https://doi.org/10.1109/wsc.1998.744909

Kelton, W. D., Smith, J. S., & Sturrock, D. T. (2014). *Simio and simulation: Modeling, analysis, Applications.* Simio LLC.

Medinilla, Á. (2016). *Agile Kaizen managing continuous improvement far beyond retrospectives*. Springer Berlin.

Sanchez, S. M. 2011. NOLHdesigns spreadsheet. Available online via http://harvest.nps.edu/ [accessed 06/01/2021]

Shahnazari-Shahrezaei, P., Tavakkoli-Moghaddam, R., & Kazemipoor, H. (2013). Solving a multi-objective multi-skilled manpower scheduling model by a fuzzy goal programming approach. *Applied Mathematical Modelling,* 37(7), 5424–5443. https://doi.org/10.1016/j.apm.2012.10.011

Viana, F. A. C. (2016). A tutorial on Latin hypercube design of experiments. *Quality and Reliability Engineering Internationa*l, 32(5), 1975–1985. https://doi.org/10.1002/qre.1924

Xie, H., Tang, H., & Liao, Y.-H. (2009). Time series prediction based on NARX neural networks: An advanced approach. 2009 *International Conference on Machine Learning and Cybernetic*s. https://doi.org/10.1109/icmlc.2009.5212326

Yildirim, S., Murat, A. E., Yildirim, M., & Arslanturk, S. (2020, September 21). Process knowledge driven change point detection for automated calibration of discrete event simulation models using machine learning. arXiv.org. https://arxiv.org/abs/2005.05385.

Zenisek, J., Holzinger, F., & Affenzeller, M. (2019). Machine learning based concept drift detection for predictive maintenance. *Computers & Industrial Engineering*, 137, 106031. https://doi.org/10.1016/j.cie.2019.10603

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California